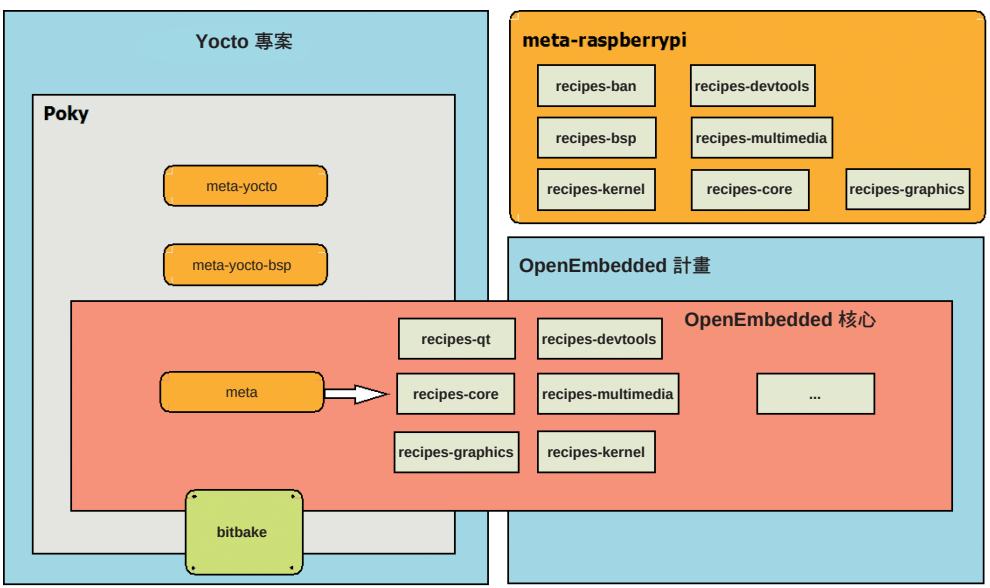
主要元件

Yocto 專案的主要元件（其他可用工具屬於選用性），條列如下：

- BitBake
- OpenEmbedded-Core
- Poky
- BSP 層（meta-raspberry、meta-fsl-arm、meta-ti、meta-intel、meta-sunxi 等等）

  譯註：BSP 是 Board Support Package 的縮寫，譯為「板子支援套件包」，內容物是各個板子專屬的組態設定、資料、規格等。

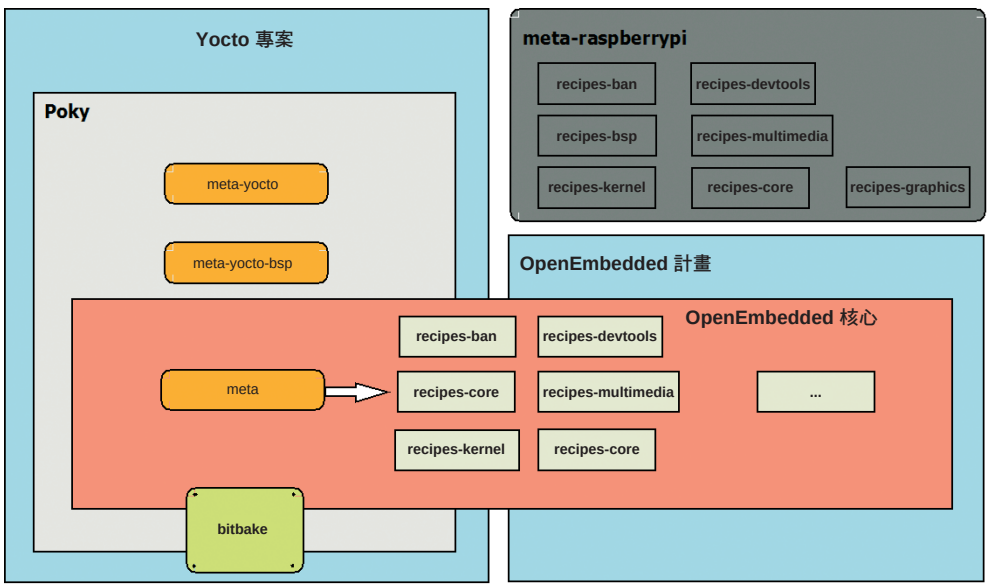
下圖秀出本書將會介紹的所有層級，後續章節將會透過各式各樣的範例，逐一研習其中諸項工具，讓讀者能充分掌握。



何謂 Poky？

Poky 是 Yocto 專案官方提供的「參考」建置系統，包含一些 OpenEmbedded 的基本元件（稱為建置系統），以及一組用來建立嵌入式發行版的後設資料，支援好幾種目標架構。Poky 是獨立的平台，使用 BitBake 工具（任務排程器）、OpenEmbedded-Core 與預設的後設資料組合來進行交叉編譯，如下圖所示。Poky 提供一整套機制，能夠建置且整合數以千計的開放原始碼計畫。

Poky 建置系統將會成為工業領域的參考標準，各大業界領袖，如 Win River、Intel、Montavista 與 Mentor Graphics，皆已表態支持。



Angstrom（`http://www.angstrom-distribution.org/`）是另一套以 OpenEmbedded-Core 為基礎所打造出來的系統，你可以把 Angstrom 與 Poky 想像成表兄弟，因為 Poky 也同樣基於 OpenEmbedded-Core。

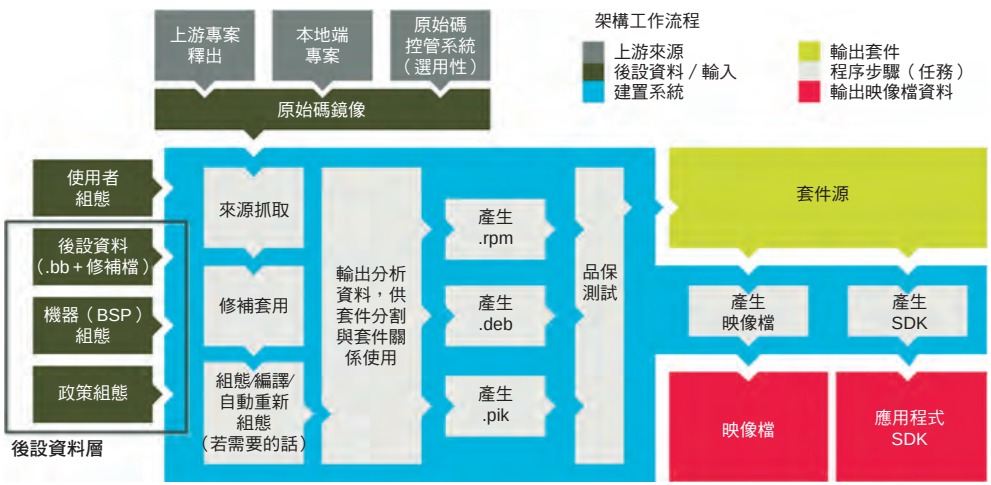
含的指令和資料，還用來指出該使用哪個版本的軟體，以及從何處取得。Poky 以 OpenEmbedded-Core 為基礎並予以擴充，加上另兩個不同的層級，屬於另外的後設資料子集合，詳細描述如下：

- **meta-yocto**：這一層提供支援的預設發行版、視覺標籤、後設資料追蹤資訊（維護者、上游狀態、等等）。
- **meta-yocto-bsp**：使用者應在這一層之上，提供硬體參考板子支援套件包（BSP），提供給 Poky。

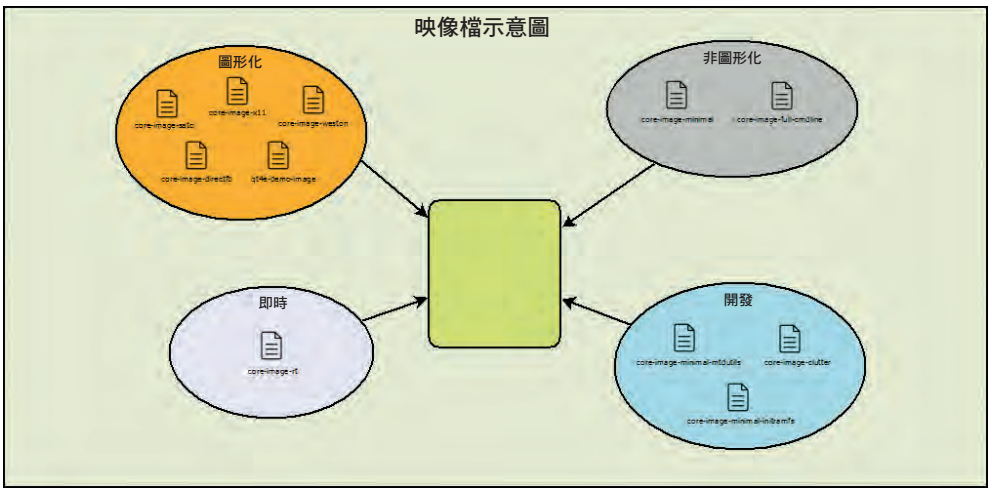
我們將在第 4 章「了解 BitBake」，深入介紹後設資料。

Yocto 專案的工作流程

下圖以較高層面的形式描繪出 Yocto 專案的開發環境，藉以呈現交叉編譯框架。



接著，下圖秀出可用映像檔的組成示意圖：



可把由 **meta-raspberry** 定義好的配方層，加入到這些層裡頭：

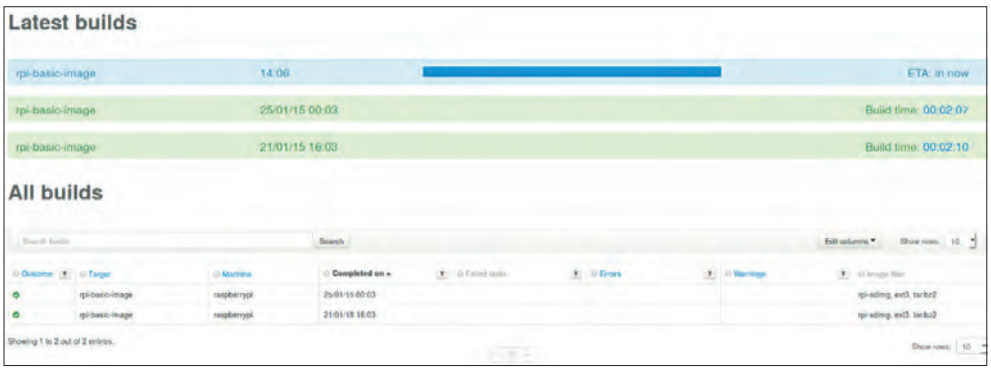
```
$ ls meta-raspberrypi/recipes-core/images/*.bb
rpi-basic-image.bb
rpi-hwup-image.bb
rpi-test-image.bb
```

- `rpi-hwup-image.bb`：這是以 core-image-minimal 為基礎的映像檔。
- `rpi-basic-image.bb`：這是以 `rpi-hwup-image.bb` 為基礎、再加上一些額外功能（如登入過程畫面）後的映像檔。
- `rpi-test-image.bb`：這是以 `rpi-basic-image.bb` 為基礎、再加上 `meta-raspberrypi` 裡某些套件的映像檔。

本章後續篇幅，將會選用這三個配方其中之一作為示範，請注意，如同其他檔案，這些檔案（`.bb`）內容描述的是配方，並以邏輯功能劃分。現在，我們擁有了適當的配方，可用來建置供 Raspberry Pi 使用的映像檔。

執行網頁介面

現在，BitBake 已啟動，我們便可監看建置程序的執行過程，如下圖所示。請注意，開啟 Toaster 介面最好的方式是以瀏覽器瀏覽（輸入網址）。



Toaster 仍處於發展階段，將來應能完美地取代掉 Hob；開發團隊希望 BitBake 能夠完全經由網頁介面來操控以及設定組態，降低使用難度。

總結

本章說明如何使用讓 BitBake 更加親民易用的介面：Hob 與 Toaster，我們學會這些工具所擁有的各項能力（組態與功能），這些工具可為開發團隊帶來更棒的作業彈性，絕對值得你一試。

下一章要說明 BitBake 在 Yocto 專案中所扮演的主要角色，我們將告訴你如何為自訂的映像檔產生出軟體套件。

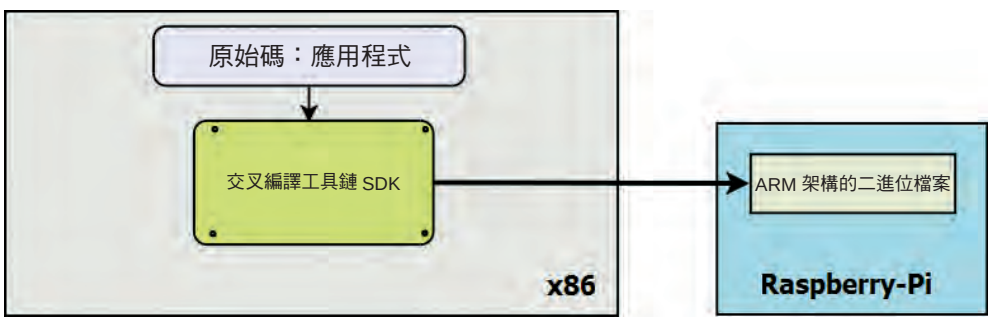
本章內容涵蓋 Yocto/OE 的基本概念，以便於整合自訂的應用程式與 Raspberry Pi，我們將會學習如何為交叉編譯應用程式產生 SDK，也會討論關於套件管理的議題。

在那之後，將會建立屬於我們自己的應用程式與配方，透過 Yocto、部署到 Raspberry Pi。

軟體開發工具組（SDK）

SDK（Software Development Kit）是可在 **Yocto/OE** 之外使用的一組工具，一般來說，包含編譯器、連結器、除錯器、以及外部標頭檔，我們將這套編譯工具組稱之為「**工具鏈**（toolchain）」。以 Raspberry Pi 來說（或其他嵌入式平台），工具鏈通常包括交叉工具組，所謂交叉工具，意思是說該工具會在某平台架構上執行，但所產生出來的二進位程式檔案，卻是用於另一個架構。

下圖是交叉編譯過程的示意圖。



Yocto/OE 建置系統可產生出交叉編譯工具鏈，並且能夠符合目標系統的 `sysroot` 目錄。

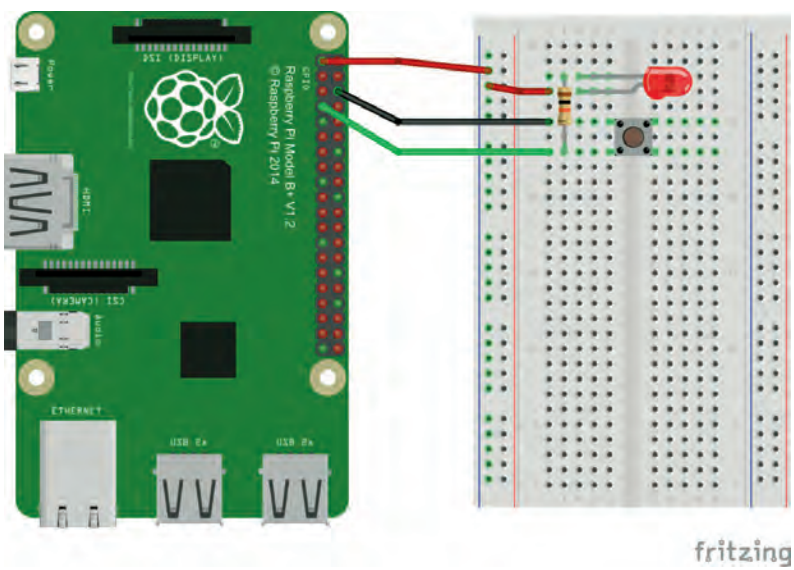
若想進一步得知詳情，還請瀏覽官方文件：
`http://www.yoctoproject.org/docs/current/dev-manual/dev-manual.html#using-runtime-package-management`。

範例應用軟體：初步介紹

現在，我們已經學會如何為目標平台產生 SDK，並且，Yocto 映像檔也能夠整合套件，接下來將要實地示範，開發一支簡單應用軟體操控 Raspberry Pi 的 **GPIO**（General Purpose Input/Output，通用輸入 / 輸出埠），然後建立配方，把這支應用軟體整合到最終的映像檔。

這支程式的功能是透過 GPIO 腳位，點亮 LED 並監看按壓式開關的狀態。

電路圖如下，以 Fritzing（`http://fritzing.org/home/`）繪製：



然後，我們將從 Linux 使用者空間，寫程式點亮 LED 或經由 Raspberry Pi 的 GPIO 4（主連接端子的針腳 7）來監看按壓開關。

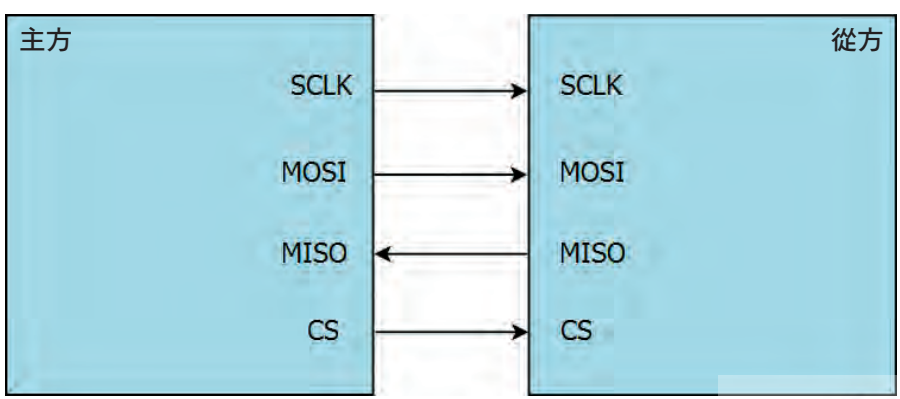
本章將學習如何透過 Yocto 來存取 Raspberry Pi 的 SPI 和 I2C 匯流排，介紹如何為自訂應用程式撰寫配方。

SPI 匯流排

SPI（Serial Peripheral Interface，序列周邊介面）協定，在主方與從方之間實作出一套同步式序列傳輸連線，若只有一個從方裝置，便只需要三條訊號線（以及接地線）即可。

SPI 協定的主方裝置負責產生時脈訊號，從 **SCLK**（serial clock）腳位輸出，送往從方裝置，在此訊號的某些上下轉變之際，從方可使用指定的訊號腳位 **MOSI**（master out slave in 的縮寫）讀取資料，或是從名為 **MISO**（master in slave out）的腳位寫出資料。製造廠商若不同，有可能賦予不同的名稱給這些腳位，建議一律採用 MISO/MOSI（最常見的命名），可免除因模糊不清造成困擾。根據 SPI 協定規格，主方的 MOSI 腳位一定要連接到從方的 MOSI 腳位，MISO 腳位也相同。

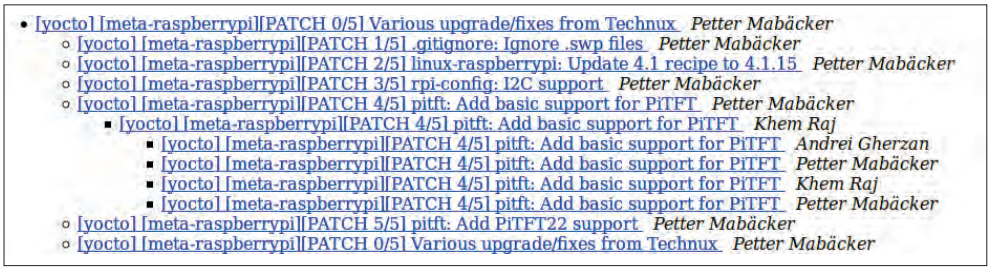
若有好幾個從方裝置要連接到同一個主方裝置，可以並聯方式連接（所有 MISO 腳位都連接在一起，MOSI 腳位也相同），但每一台從方裝置都需要獨立的選擇訊號線路（**CS**，Chip Select），才能決定在某一時間可由哪個從方裝置收發資料。示意圖如下：



PiTFT 2.8 吋電阻式觸控螢幕執行 `core-image-sato`

Raspberry Pi 開機時，LCD 會先顯示白色畫面，稍等一會後又再轉成黑色，代表核心已成功辨認這台螢幕，過沒多久，應該就會在螢幕上看到視窗系統的畫面。2.8 吋螢幕相當小，但仍可使用虛擬鍵盤輸入些文字，譬如在終端機裡輸入指令。這台觸控螢幕已經使用預設值進行校準，可用於顯示和旋轉，若你覺得預設校準的結果仍不滿意，譬如想要修改螢幕的擺向（預設為 90 度），你可以使用 `ts_calibrate` 重新校準。

`ts_calibrate` 是 Poky 的 `tslib` 配方的一部份，預設會被放進 `building core-image-sato`，只要指定 `MACHINE_FEATURES += "touchscreen"`，就能輕易地放進你的自訂映像檔。



只要註冊特定的郵遞論壇，就能輕鬆追蹤最新動態，另外也可以自行到備份區翻閱舊記錄。

Yocto 所有的郵遞論壇，可到以下網址查詢：`https://www.yoctoproject.org/tools-resources/community/mailing-lists`。

貢獻給 meta-raspberrypi

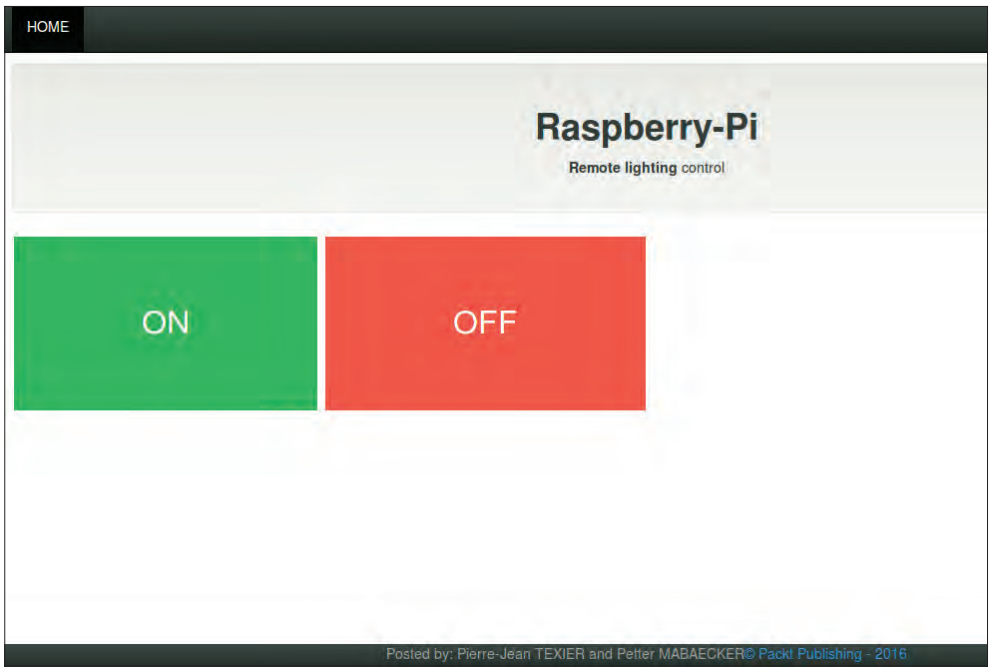
若你有新功能、或是修正了某個臭蟲，想要貢獻給 meta-raspberrypi，與整個社群分享，必須先了解幾件事情。

meta-raspberrypi 目前維護者是 Andrei Gherzan，對於他人提交的修補檔，他擁有最後的決定權，並且會把修補檔整合到 meta-raspberrypi 主層裡。維護者們都很忙碌，因此你必須小心謹慎地測試修改的地方，確認無誤再送往上游，這點非常重要。對於一名貢獻者來說，不論參與哪項專案，最重要的事情是建立起好名聲，根據專案要求的規定，只寄出經過良好測試的修補檔，這麼做的話，修補檔被接受並整合到專案裡的機會，必然大幅提昇。

貢獻給 meta-raspberrypi 時，有幾項概略「守則」：

- 與 meta-raspberrypi 相關的修補檔，必須寄給 `yocto@yoctoproject.org`。
- 在郵件主題前加上「**[meta-raspberrypi]**」。

若以瀏覽器開啟 index.html，可看到如下畫面：



客戶端（index.html）顯示於網頁瀏覽器中的模樣

最後，可別忘了把 gpio-packt 和 webserver-packt 加入映像檔：

```
$ cat recipes-core/images/packt-iot-image.bb
# Base this image on rpi-basic-image
include recipes-core/images/rpi-basic-image.bb
SPLASH = "psplash-raspberrypi"
IMAGE_FEATURES += "ssh-server-dropbear splash"
IMAGE_INSTALL_append = " rpi-gpio gpio-packt webserver-packt"
```

若因為某種原因，伺服器並未自動啟動，可試著輸入底下指令，手動啟動：

```
$ /etc/init.d/server-packt-init start
starting Nodejs app: server.init... done.
root@raspberrypi2:~#    info  - socket.io started
listening on *:3344
```

若直接在 Raspberry Pi 上對網站伺服器進行修改，可使用底下指令重新啟動：

```
$ /etc/init.d/packt-server restart
stopping Nodejs app: server.init... stopped node (pid 295)
done.
starting Nodejs app: server.init... done.
```

現在，請開啟智慧型手機的瀏覽器，輸入網址 `http://my_rpi_ipaddress:3344`，看到的網頁，應該就跟本章先前開啟 `index.html` 看到的畫面一樣，其格式或許因為手機螢幕尺寸而稍有不同，但一定會有紅色與綠色按鈕，看到了嗎！請試著點按按鈕，便可看到檯燈隨之開關。

數位家庭自動化專案的最終成品

至此，你的數位家庭自動化專案已經完成且動起來了，邁向無線連接的居家環境，你又往前進了一步，希望本章專案會激發你想出各種點子，運用 Raspberry Pi 製作更有趣更有創意的專案。